Payment Handler API

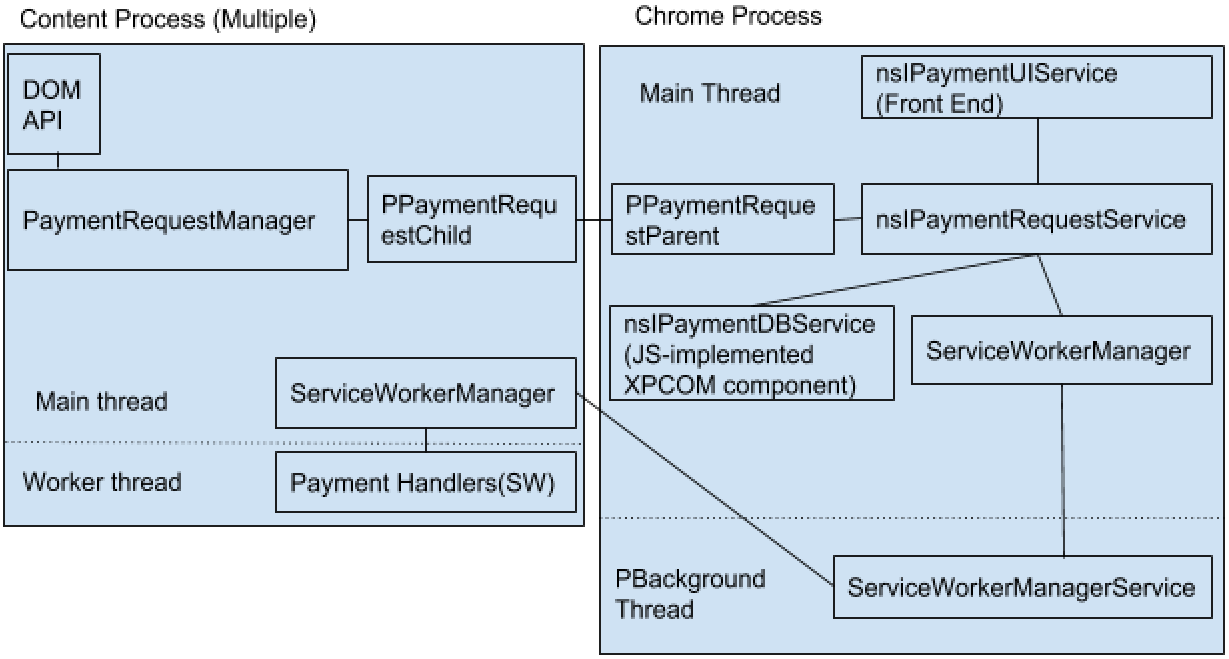
Introduction

Flow:

1. Users grant permission to origins which provides payment handlers, such as retail or bank sites.
2. Payment handlers are registered via service worker registration, lists of enabled payment methods and capabilities of payment handlers are specified using paymentManager attribute in ServiceWorkerRegistration interface.
3. When PaymentRequest::show() is called, UA matches and displays a list of payment instruments which are able to handle this payment request.
4. After an instrument is selected by the user, a PaymentRequestEvent is fired in the target payment handler (service worker) whose PaymentManager the instrument was registered with.
5. The payment handler processes the request and provides the response to UA using PaymentRequestEvent::respondWith().

Ref: https://www.w3.org/TR/payment-handler/#model

Proposed Architecture



Why main thread bounded instead of using background thread?

- Frequent interaction with browser UI but limited interaction (PaymentRequestEvent + PaymentResponse) with SWs
- A bit simpler architecture on the storage part using JS-implemented XPCOM component to utilize IndexDB directly (Push API impl. for ref.)

Implementation Subtasks:

1. **nsIPaymentDBService:** JS-Implemented XPCOM component to use indexDB for storing instrument records
    - PushService.jsm
    - PushDB.jsm
2. **ServiceWorkerRegistration::PaymentManager**
    - Implemented by both ServiceWorkerRegistrationMainThread and ServiceWorkerRegistrationWorkerThread
    - PaymentInstruments: API for interacting with the local storage (through nsIPaymentDBService) such as set/get instrument records
3. **PaymentRequest.show()**: For each paymentMethod in the request, determine which payment handlers support this method.
    - Implement a matching algorithm which interacts with nsIPaymentDBService to find a candidate list of payment handlers
    - Interact with browser UI to show candidates for user to choose
    - Trigger the dispatch of PaymentRequestEvent to selected payment handlers
4. PaymentRequestEvent
    - Dispatch a PaymentRequestEvent triggered by PaymentRequest.show() to the payment handler chosen by user
        - WIP Patches
    - Provide UA payment responses by PaymentRequestEvent::respondWith()
        - Spec Pull Request: Define PaymentRequestEvent.respondWith() behavior.
    - Implement PaymentRequestEvent::openWindow()
        - Spec Issue: Open Window Algorithm

Reference

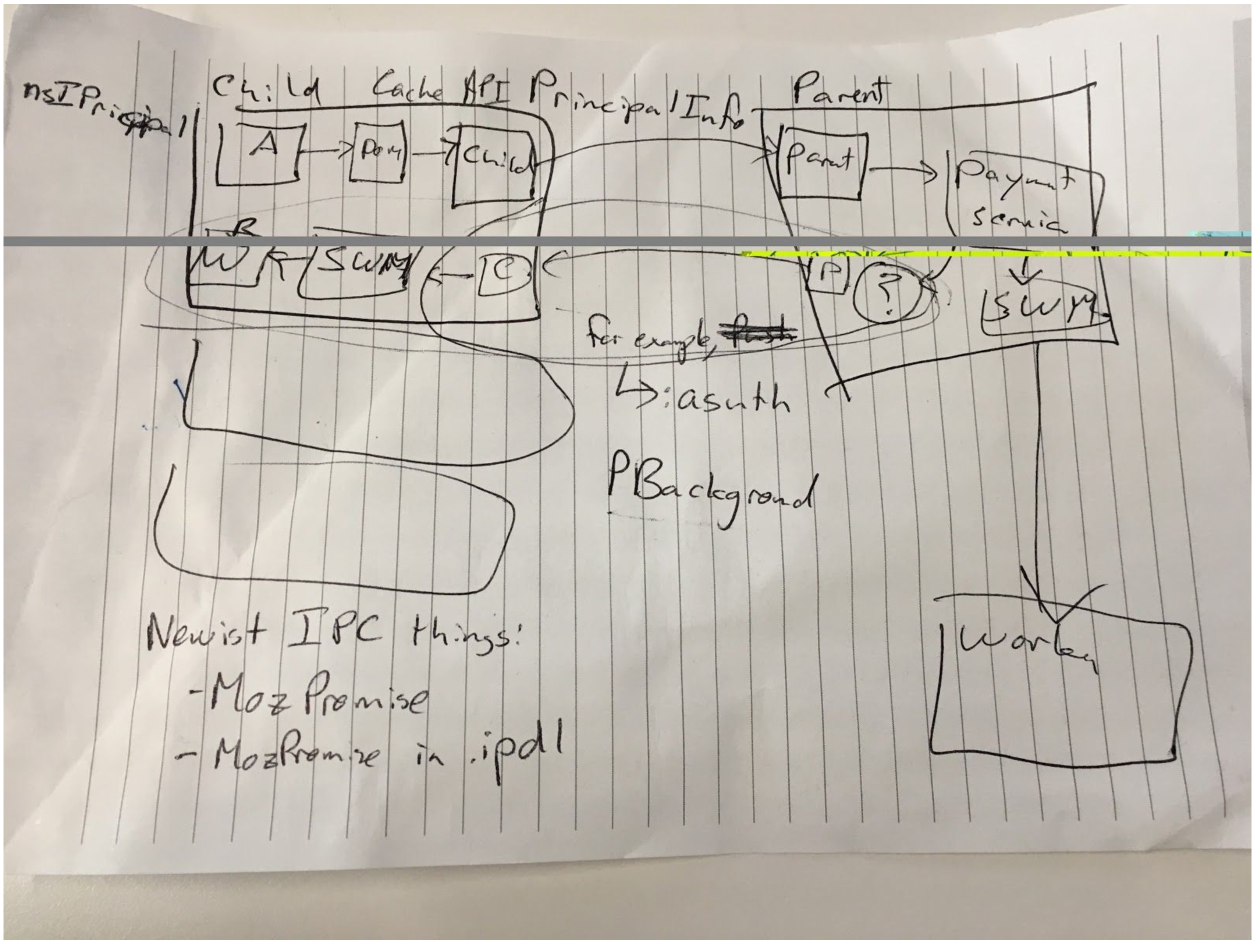
[1] API for accessing the target SW from SWMS: https://bugzilla.mozilla.org/show_bug.cgi?id=1368625

[2] https://www.w3.org/TR/payment-handler/

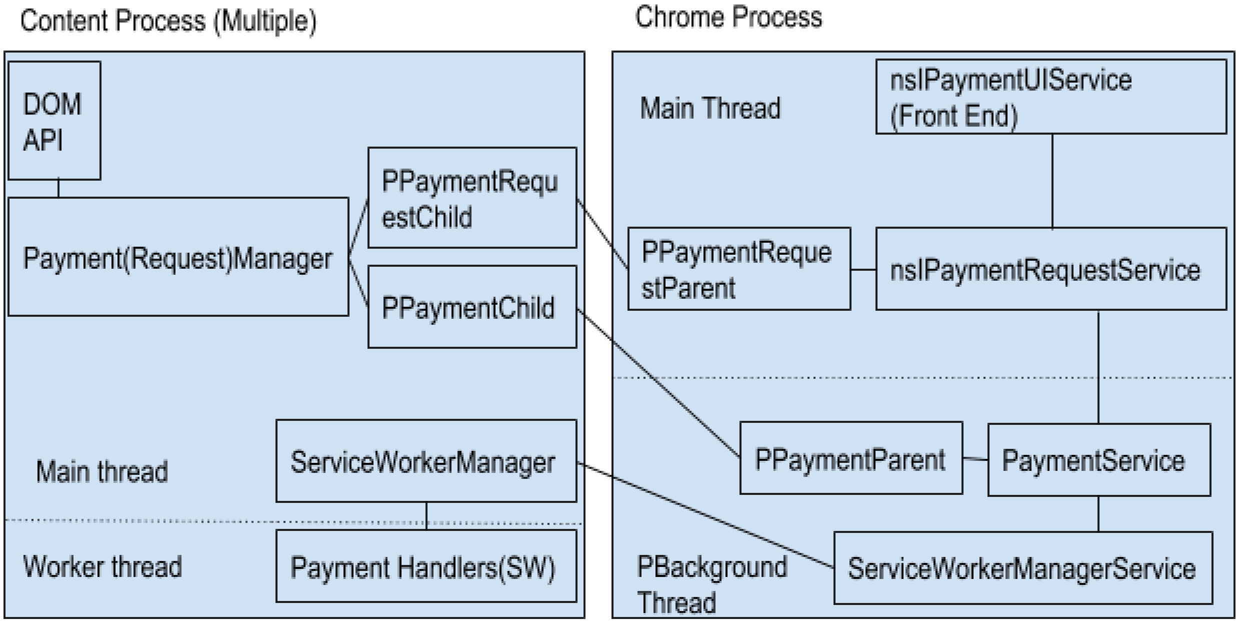
[3] BackgroundSync API: https://bugzilla.mozilla.org/show_bug.cgi?id=1217544

[4] Cache API: https://blog.wanderview.com/blog/2014/12/08/implementing-the-serviceworker-cache-api-in-gecko/

[5] Suggested architecture draft from bkelly



[6] Proposed Architecture using PBackground



Why PBackground?

- To be complied with other service worker related API, such as cache API and background sync API, and be easier to fit into the new architecture of SW.
- Easier to support non-e10s mode in the future by migrating PBrowser to PBackground.
- Move IPC overhead from main thread of chrome process to PBackground thread.
- Easier to communicate with storages in the future using QuotaClient + SQLite/mozStorage (Cache and backgroundsync API impl. for ref.)